# Bachelorabschlussarbeit
# Parallelisierte Faktorisierung mit dem Quadratischen Sieb

Erstellt von Georg Hahn

Erscheinungsdatum: 08 Mai 2008

Mathematisches Institut
Johannes Gutenberg Universität Mainz

# Inhaltsverzeichnis

**1 Einleitung** **3**

**2 Der Basisalgorithmus des Quadratischen Siebes** **4**
2.1 Faktorisierung nach Fermat . . . . . 4
2.2 Von der Faktorisierung nach Fermat zum Quadratischen Sieb . . 5
2.3 Algorithmische Umsetzung . . . . . 6
2.3.1 Die Wahl einer Faktorbasis . . . . . 6
2.3.2 Die Kombination zu einem Quadrat . . . . . 7
2.4 Das Quadratische Sieb . . . . . 9
2.4.1 Initialisierung . . . . . 9
2.4.2 Sieben der Kongruenzen . . . . . 10
2.4.3 Praktische Umsetzung des Siebschrittes . . . . . 12
2.4.4 Kombination zu einer Darstellung $a^2 \equiv b^2 \pmod{n}$ . . . . 13
2.4.5 Ermittlung der Faktoren per ggT . . . . . 14
2.5 Large Prime Variation . . . . . 16
2.6 Computeranwendung und Parallelisierung . . . . . 18

**3 Realisierung eines Netzwerks zur parallelisierten Faktorisierung** **19**
3.1 Das Serverprogramm . . . . . 19
3.2 Der Siebschritt . . . . . 21
3.3 Kernberechnung und Faktorisierung . . . . . 23
3.4 Die Java-Anwendung . . . . . 24

**4 Theoretische Laufzeit und Beispiele paralleler Faktorisierung** **25**
4.1 Siebmethoden . . . . . 25
4.2 Legendre-Symbol . . . . . 25
4.3 Zentrierte Siebintervalle um die Wurzel . . . . . 26
4.4 Large Prime Variation . . . . . 27
4.5 Ergebnisse der Faktorisierung mit einem Client . . . . . 28
4.6 Parallelisierung . . . . . 28
4.7 Theoretische Laufzeit . . . . . 30

**5 Diskussion der Ergebnisse** **31**

**6 Ausblick auf Erweiterungen des Basisalgorithmus des Quadratischen Siebes** **32**

**7 Eigenständigkeitserklärung** **33**

# 1 Einleitung

Seit Euklid um 300 v. Chr. bewies, dass jede natürliche Zahl eine eindeutige Darstellung in Primfaktoren besitzt, ist die Faktorisierung natürlicher Zahlen ein Forschungsgebiet der Mathematik. Allerdings blieb die sogenannte Probedivision („Trial Division“), d.h. das Probieren aller Teiler bis zur Wurzel der zu faktorisierenden Zahl, bis zum 17. Jahrhundert die einzige bekannte Faktorisierungsmethode. 1643 formulierte Pierre de Fermat ein neues Faktorisierungsverfahren, das eine Zahl als nicht triviale Differenz zweier Quadrate darstellt. Aus einer solchen Darstellung lassen sich anschließend zwei Teiler, die nicht notwendig Primzahlen sein müssen, berechnen. Diese Technik wird im folgenden Kapitel dargestellt und liefert zudem die Grundlage für weitere Faktorisierungsverfahren wie der Kettenbruchmethode von Morrison-Brillhart, dem Quadratischen Sieb und dem Zahlkörpersieb sowie deren Varianten.

Das Quadratische Sieb wurde 1981 von Carl Pomerance vorgestellt und basiert auf der von Fermat eingeführten Strategie, eine Quadratdarstellung zu konstruieren. Es baut auf vorherige Ansätze von Kraitchik and Dixon auf. Das Verfahren war bis 1993 die schnellste bekannte Faktorisierungsmethode, bis es durch eine Weiterentwicklung, dem Zahlkörpersieb („Number Field Sieve“ NFS) abgelöst wurde, das bis heute als schnellstes Verfahren gilt. Dennoch zeigt sich, dass das Quadratische Sieb Zahlen bis ca. 110 Stellen in der Praxis am schnellsten faktorisiert.

Im zweiten Kapitel der Arbeit wird die Theorie der Basisversion des Quadratischen Siebes vorgestellt. Es folgen Details zur algorithmischen Umsetzung und speziell der Bezug auf den parallelisierten Einsatz in der Praxis. Dazu wurde der Algorithmus im Rahmen dieser Arbeit als Server-Client-Netzwerk realisiert, dessen Details der Implementierung im 3. Kapitel vorgestellt werden. Es folgen eine theoretische Laufzeitbetrachtung sowie Testfälle des parallelisierten Einsatzes und eine anschließende Diskussion der Ergebnisse im 4. und 5. Kapitel. Die Arbeit schließt mit einem Ausblick auf Erweiterungen des Basisalgorithmus des Quadratischen Siebes im 6. Kapitel.

# 2 Der Basisalgorithmus des Quadratischen Siebes

Das Quadratische Sieb beruht auf der Darstellung der zu faktorisierenden Zahl als Differenz zweier Quadrate. Diese Idee geht auf Pierre de Fermat zurück und wird im Folgenden beschrieben.

## 2.1 Faktorisierung nach Fermat

Ist für eine zu faktorisierende Zahl $n \in \mathbb{N}$ eine Darstellung als Differenz zweier Quadrate gefunden, d.h.

$$n = a^2 - b^2$$

mit $a, b \in \mathbb{N}$, so folgt nach der dritten Binomischen Formel direkt

$$n = (a + b)(a - b).$$

Gilt $(a + b) > 1$ und $(a - b) > 1$, so ist diese Faktorisierung nicht trivial. Umgekehrt besitzt jede zusammengesetzte Zahl $n = xy$ eine solche Zerlegung. Dies folgt mit

$$a := \frac{1}{2}(x + y),$$

$$b := \frac{1}{2}|x - y|.$$

Um nun eine Zahl zu faktorisieren, gibt man einen Startwert für $a \in \mathbb{N}$ vor, berechnet $a^2 - n$ und testet, ob das Ergebnis wieder eine Quadratzahl ist. Dabei startet man mit der ersten ganzen Zahl $a \geq \sqrt{n}$, da ansonsten auf Grund von $a^2 - n < 0$ eine Darstellung in positiven Quadraten unmöglich ist, und erhöht $a$ in jeder Iteration um 1. Ist ein $b \in \mathbb{N}$ mit $a^2 - n = b^2$ gefunden, so lässt sich $n$ wie oben beschrieben faktorisieren.

Die Methode soll am Beispiel $n = 3007$ demonstriert werden. Beginnend mit $a = \lceil \sqrt{3007} \rceil = 55$ wird

$$\begin{aligned} 55^2 - 3007 &= 18 \\ 56^2 - 3007 &= 129 \\ &\vdots \\ 64^2 - 3007 &= 1089 = 33^2 \end{aligned}$$

berechnet, bis schließlich mit $a = 64$ die Faktorisierung

$$3007 = 64^2 - 33^2 = (64 + 33)(64 - 33) = 97 * 31$$

gefunden wird.

Für $n$ gerade (genauer: $n = 2t$; $t \in \mathbb{N}$ ungerade) ist die Methode nicht anwendbar, da dann $a = \frac{1}{2}(x + y) \notin \mathbb{N}$ ist. Da für gerade Zahlen direkt alle Primfaktoren 2 angegeben werden können, bis die Zahl auf einen ungeraden Rest reduziert ist, wird das Problem der Faktorisierung auf ungerade Zahlen bezogen.

Die Technik von Fermat findet dann schnell einen Teiler, falls $x$ und $y$ möglichst nahe beieinander liegen, da in diesem Fall $\sqrt{n} \approx x \approx y \approx a$ gilt

und beginnend ab $\sqrt{n}$ schnell eine Quadratdarstellung gefunden wird. Dies ist auch der Fall, für den eine Testdivision mit Startwert 2 am längsten benötigt. Umgekehrt muss der Startwert $a \geq \sqrt{n}$ der Fermat-Methode über viele Iterationen erhöht werden, falls die zwei Faktoren $x$ und $y$ von $n$ weit auseinander liegen. Der „Worst-Case“ mit ungeradem $n$ tritt damit für $n = 3p$ mit $p$ prim ein, da der gesuchte Wert $a = \frac{1}{2}(3 + p)$ dann am weitesten von der Wurzel entfernt liegt. In diesem Fall ist

$$a = \frac{1}{2}(3 + p) = \frac{1}{2}(3 + \frac{n}{3}) = \frac{1}{6}(n + 9).$$

Der Algorithmus im Pseudocode lautet:

*Faktorisierung nach Fermat*

for $a = \lceil \sqrt{n} \rceil$ to $\frac{1}{6}(n + 9)$ do
    if($b = \sqrt{a^2 - n} \in \mathbb{N}$) return($n = (a + b)(a - b)$);
return($n$ ist prim);

Eine Analyse der Fermat-Methode zeigt nach [2], dass für einen „Standardfall“ mit zwei Faktoren der Größenordnung $x \approx n^{\frac{1}{3}}$ und $y \approx n^{\frac{2}{3}}$ der Algorithmus ca. $\frac{1}{2}n^{\frac{2}{3}}$ Schritte zur Faktorisierung benötigt und damit signifikant langsamer als die Probedivision mit Laufzeit $O(n^{\frac{1}{2}})$ ist.

## 2.2 Von der Faktorisierung nach Fermat zum Quadratischen Sieb

Im Quadratischen Sieb wird nun die Suche nach $a$, $b \in \mathbb{N}$ mit $n = a^2 - b^2$ verallgemeinert. Gesucht werden $a$, $b \in \mathbb{N}$ mit

$$a^2 \equiv b^2 \pmod{n}. \tag{1}$$

Falls zusätzlich

$$a \not\equiv \pm b \pmod{n} \tag{2}$$

erfüllt ist, gilt $n \mid (a+b)(a-b)$ nach (1), jedoch $n \nmid (a+b)$ und $n \nmid (a-b)$ nach (2). Falls diese zwei Bedingungen zutreffen, beinhaltet $(a+b)$ mindestens einen Primfaktor von $n$, der in $(a-b)$ nicht enthalten ist und umgekehrt. Mit Hilfe des größten gemeinsamen Teilers erhält man daher eine Faktorisierung in

$$1 < gcd(n, a + b) < n \text{ und } 1 < gcd(n, |a - b|) < n,$$

wobei beide *gcd* nicht trivial sind, da $(a + b)$ und $(a - b)$ jeweils mindestens einen unterschiedlichen Primfaktor von $n$ enthalten und beide nicht von $n$ geteilt werden. Allerdings sind beide Faktoren nicht notwendig prim.

Um eine Darstellung (1) zu erhalten, wird nun versucht, einzelne Kongruenzen, die selbst keine Quadratdarstellung von $n$ erlauben, zu einer solchen zu kombinieren. Im Beispiel der Faktorisierung von $n = 3007$ aus Kapitel 2.1 ist dies folgendermaßen möglich: Berechnet wurde

$$\begin{aligned} 55^2 &\equiv 2 * 3^2 \pmod{3007} \\ 56^2 &\equiv 3 * 43 \pmod{3007} \\ 57^2 &\equiv 2 * 11^2 \pmod{3007} \end{aligned}$$

bis $64^2$, womit die Faktorisierung von 3007 gefunden wurde. Durch Kombination der obigen Kongruenzen kann eine Faktorisierung bereits nach drei faktorisierten Kongruenzen gefunden werden, nämlich mit Hilfe der ersten und dritten:

$$55^2 * 57^2 \equiv 3135^2 \equiv 2 * 3^2 * 2 * 11^2 \equiv 2^2 * 3^2 * 11^2 \equiv 66^2 \pmod{3007}.$$

Daraus erhält man

$$3135^2 \equiv 66^2 \pmod{3007},$$

$$3135 \equiv 128 \not\equiv \pm 66 \pmod{3007},$$

sodass per $gcd(3007, 128+66) = 97$ und $gcd(3007, 128-66) = 31$ die Faktorisierung von 3007 gefunden wird. In diesem Beispiel erfüllt die Quadratdarstellung zusätzlich die Bedingung (2), sodass die Faktorisierung folgt.

Gilt (2) nicht, so erhält man wegen $gcd(n, a-b) = gcd(n, 0) = n$ bzw. $gcd(n, a+b) = gcd(n, 0) = n$ eine triviale Faktorisierung.

## 2.3 Algorithmische Umsetzung

Das im Beispiel in Kapitel 2.2 vorgestellte Verfahren wurde ohne Berücksichtigung algorithmischer Details vorgestellt. Daher soll nun präzisiert werden, wie beispielsweise die Faktorisierung der Kongruenzen oder die algorithmische Bestimmung einer geeigneten Auswahl der Kongruenzen zur Bildung von Quadraten genau umgesetzt wird.

Zuvor wird die Notation festgelegt, die in den folgenden Erläuterungen benutzt wird, wobei alle Variablen natürliche Zahlen darstellen. Mit $n$ wird die zu faktorisierende Zahl bezeichnet, die Faktorisierung sei $n = xy$. Die ab der Wurzel betrachteten Quadrate seien $a_i^2$. Gesucht ist eine Darstellung $a^2 \equiv b^2 \pmod{n}$.

### 2.3.1 Die Wahl einer Faktorbasis

Am Beispiel unter 2.2 ist zu sehen, dass nicht alle faktorisierten Kongruenzen später zur Kombination der Quadratdarstellung benutzt wurden. Daher ist es nicht sinnvoll, alle Kongruenzen vollständig zu faktorisieren, da dies zu viel Zeit in Anspruch nimmt. Beispielsweise lautete die zweite Kongruenz $56^2 \equiv 3 * 43 \pmod{3007}$, in der ein relativ großer Primfaktor (43) in ungerader Potenz auftrat und es somit sehr unwahrscheinlich war, wieder eine Kongruenz mit Faktor 43 zu finden, um beide zu einem Quadrat zu kombinieren. Tatsächlich wurde diese Kongruenz zur anschließenden Faktorisierung von 3007 auch nicht genutzt.

Aus diesem Grund schränkt man die Faktorisierung der Kongruenzen auf alle Primfaktoren kleiner einer zuvor gewählten Schranke $B$ ein. Als Faktorbasis

$$F = \{p_1, p_2, \ldots, p_k\}$$

bezeichnet man die Menge aller zur Faktorisierung der Kongruenzen benutzen Primzahlen. Eine Zahl, die innerhalb einer Faktorbasis mit Primzahlen bis zur Schranke $B$ („*Smoothness Bound*“) zerfällt, d.h. nur Primfaktoren aus $F$ besitzt, wird als „*B-smooth*“ oder „*B-glatt*“ bezeichnet. Die Faktorisierung der Kongruenzen im Quadratischen Sieb wird dabei entweder per Testdivision oder mit Hilfe eines exponentiellen Faktorisierungsalgorithmus wie beispielsweise Pollard-Rho oder Pollard-(p-1) durchgeführt.

### 2.3.2 Die Kombination zu einem Quadrat

Ist eine gewisse Menge an faktorisierten $B$-glatten Kongruenzen, die über einer Faktorbasis

$$F = \{p_1, p_2, \ldots, p_k\}$$

zerfallen sind, gefunden, so ist es essentiell für die Faktorisierung von $n$, eine geeignete Auswahl genau der Kongruenzen $a_i^2$ mit $i \in I$, $I$ Indexmenge, zu finden, deren Produkt eine Quadratdarstellung $a^2 \equiv b^2 \pmod{n}$ ermöglicht. Es seien $r$ Kongruenzen der folgenden Form gefunden worden:

$$\begin{aligned} a_1^2 &\equiv p_1^{e_{1,1}} \cdots p_k^{e_{k,1}} \pmod{n} \\ &\vdots \\ a_r^2 &\equiv p_1^{e_{1,r}} \cdots p_k^{e_{k,r}} \pmod{n} \end{aligned}$$

Gesucht ist ein Produkt

$$\prod_{i \in I} a_i^2 \equiv p_1^{d_1} \cdots p_k^{d_k} \pmod{n},$$

das ausschließlich gerade Exponenten $d_j = \sum_{i \in I} e_{j,i}$ enthält. In diesem Fall folgt eine Quadratdarstellung

$$a^2 := \left( \prod_{i \in I} a_i \right)^2 \equiv \left( p_1^{\frac{1}{2} d_1} \cdots p_k^{\frac{1}{2} d_k} \right)^2 =: b^2 \pmod{n}$$

aus der per ggT eine Faktorisierung von $n$ berechnet werden kann, falls zusätzlich

$$a \not\equiv b \pmod{n}$$

erfüllt ist.

Da die eigentlichen Werte der gesuchten Exponenten $d_j$ keine Rolle spielen, sondern diese nur gerade sein müssen, lässt sich diese Bedingung zu

$$d_j \equiv 0 \pmod{2}$$

vereinfachen. Die Exponenten der einzelnen Kongruenzen $a_i^2$ werden zudem als Spaltenvektor

$$v_i = \begin{pmatrix} e_{1,i} \\ \vdots \\ e_{k,i} \end{pmatrix}$$

umgeschrieben, sodass nun die Suche nach einer Auswahl $I$ der Kongruenzen mit geraden Primzahlexponenten im Produkt der Suche nach einer Lösung von

$$\sum_{i=1}^{r} c_i * v_i \equiv 0 \pmod{2} \quad \text{mit } c_i \in \mathbb{F}_2$$

entspricht. Dabei geben die Koeffizienten $c_i \in \mathbb{F}_2$ für jede Kongruenz $a_i^2$ an, ob diese im Produkt benötigt wird ($c_i = 1$) oder nicht ($c_i = 0$).

Werden die Exponentenvektoren $v_i$ in einer Matrix

$$M = \begin{pmatrix} \vdots & & \vdots \\ v_1 & \cdots & v_r \\ \vdots & & \vdots \end{pmatrix}$$

zusammengefasst, so ist das gesamte Problem einer geeigneteten Auswahl der Kongruenzen auf eine Kernberechnung der Matrix $M$ über $\mathbb{F}_2$ zurückgeführt worden. Dafür können Verfahren wie der Gauß-Algorithmus benutzt werden. In einem Basisvektor des Kerns stehen dann die obigen $c_i$, sodass pro Kernvektor das Produkt aller $a_i^2$, für die im Eintrag $i$ eine 1 steht, nur gerade Exponenten der Primfaktoren besitzt. Man erhält somit pro Kernvektor eine Darstellung $a^2 \equiv b^2 \pmod{n}$, die eine Faktorisierung von $n$ ermöglicht, sofern sie nicht trivial ist.

Der Vorteil dieser Technik ist zudem, dass genau angegeben werden kann, wieviele Kongruenzen $a_i$ gefunden werden müssen, bis ein nicht trivialer Kern der Matrix garantiert ist. Besitzt die Matrix nämlich mehr Spalten als Zeilen, so muss sie singulär sein. Dies bedeutet, dass mindestens $k+1$ Kongruenzen gefunden werden müssen, da die Exponentenvektoren $k$ Einträge besitzen und als Spalten in die Matrix eingetragen werden. Somit entspricht die Zeilenanzahl der Exponentenmatrix der Anzahl an Primzahlen in der Faktorbasis und die Spaltenanzahl der Gesamtzahl gefundener Kongruenzen. Die Anzahl der zu ermittelnden Kongruenzen hängt damit direkt von der gewählten Schranke $B$ ab.

Am Beispiel aus 2.2 soll die Faktorisierung nun per Kernvektoren bestimmt werden. Dazu werden die gefundenen Kongruenzen

$$\begin{aligned} 55^2 &\equiv 2 * 3^2 \pmod{3007} \\ 56^2 &\equiv 3 * 43 \pmod{3007} \\ 57^2 &\equiv 2 * 11^2 \pmod{3007} \end{aligned}$$

nun als Spaltenvektoren (mod 2) zur Faktorbasis $F := \{2, 3, 11, 43\}$ in eine Matrix umgeschrieben. Die erste Kongruenz

$$55^2 \equiv 2^1 * 3^2 * 11^0 * 43^0 \pmod{3007}$$

entspricht daher dem ersten Spaltenvektor $(1, 0, 0, 0)^T$, sodass sich schließlich die Matrix

$$M = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

ergibt. Nach der Kernberechnung erhält man

$$Ker(M) = \left\langle \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right\rangle$$

und damit die bereits bekannte Faktorisierung mit Hilfe der ersten und dritten Kongruenz. Im Beispiel besitzt die Matrix allerdings noch mehr Zeilen als Spalten, sodass ein nicht trivialer Kern nicht garantiert war. In der Praxis wäre die

Kernberechnung im „Matrixschritt“ somit erst nach mindestens zwei weiteren gefundenen Kongruenzen ausgeführt worden.

Im Beispiel ist zu sehen, dass die Matrix mehr Nullen als Einsen enthält. Für größere Matrizen nimmt dieses Ungleichgewicht noch weiter zu, da eine faktorisierte Kongruenz meist nur wenige Faktoren der Faktorbasis enthält und somit die größeren Matrizen überwiegend aus Nullen bestehen. Solche Matrizen werden als „sparse matrix“ bezeichnet. Es ist möglich, eine solche Matrix nur mit ihren 1-Einträgen abzuspeichern und spezielle schnelle Algorithmen wie beispielsweise die Methoden von Lanczos [8] oder Wiedemann [9] statt des Gauß-Algorithmus zu verwenden.

## 2.4 Das Quadratische Sieb

Die Idee, verschiedene Kongruenzen zu einer Darstellung $a^2 \equiv b^2 \pmod{n}$ zu kombinieren, geht auf Maurice Kraitchik (um 1920) zurück. Das Quadratische Sieb benutzt diese Methode und ist eine Weiterentwicklung des Dixon Algorithmus (John Dixon 1981), der wie das Quadratische Sieb $a_i^2 \pmod{n}$ faktorisiert, diese aber randomisiert und nicht aufsteigend ab $\sqrt{n}$ wählt.

Der gesamte Algorithmus zur Faktorisierung eines $n \in \mathbb{N}$ lässt sich dabei nach [1] in die folgenden Phasen einteilen, die nun näher beschrieben werden:

- Initialisierung
- Sieben der Kongruenzen
- Kombination zu einer Darstellung $a^2 \equiv b^2 \pmod{n}$
- Ermittlung der Faktoren per ggT

### 2.4.1 Initialisierung

Im vorherigen Kapitel wurde die partielle Faktorisierung bis zu einer Schranke $B$ beschrieben, um die Suche nach Kongruenzen zu beschleunigen. Wird $B$ dabei zu klein gewählt, so enthält die Faktorbasis nur wenige Primzahlen und es werden nur sehr wenige Kongruenzen gefunden, die vollständig über der Faktorbasis zerfallen. Dafür werden nach 2.3.2 auch nur wenige Kongruenzen benötigt, um eine lineare Abhängigkeit der Exponentenmatrix zu garantieren. Bei zu großem Wert von $B$ zerfallen zwar viele Kongruenzen über der großen Faktorbasis, dafür müssen andererseits auch sehr viele Kongruenzen gefunden werden, bis der Kern der Exponentenmatrix berechnet werden kann. Daher ist ein von der zu faktorisierenden Zahl abhängiger Wert für $B$ sinnvoll. Nach [1] hat sich als heuristischer Wert

$$B \approx \exp\left(\frac{1}{2}\sqrt{\ln n \ln\ln n}\right)$$

bewährt. Die Länge des Siebintervalls, das durchsucht werden muss, um genügend $B$-glatte Kongruenzen zu finden, beträgt dann

$$L \approx B^2.$$

Mit „Sieben der Kongruenzen“ bezeichnet man das Durchsuchen des Siebintervalls und die Faktorisierung der $a_i^2 \pmod{n}$, bis genügend $B$-glatte Kongruenzen

zum Eintragen in die Matrix gefunden wurden. Dieser Schritt des Algorithmus wird auch „Siebschritt“ genannt. Die Faktorbasis $F$ wird somit mit Primzahlen bis zur Schranke $B$ initialisiert, wobei zu beachten ist, dass für eine Primzahl $p \leq B$ mit

$$p \mid (a_i^2 - n)$$

auch gilt, dass

$$a_i^2 \equiv n \pmod{p},$$

sodass nur die Primzahlen bis zur Schranke $B$ in $F$ aufgenommen werden, für die $n$ ein quadratischer Rest ist. Es wird daher geprüft, ob für das Legendre-Symbol

$$\left(\frac{n}{p}\right) = 1$$

zutrifft, andernfalls tritt $p$ niemals als Faktor eines $a_i^2 \pmod{n}$ auf und muss nicht in die Faktorbasis aufgenommen werden. Da für eine Primzahl $p$ ca. die Hälfte aller $r \in \{0, \ldots, p-1\}$ quadratische Reste sind, wird durch Überprüfen des Legendre-Symbols die Anzahl der Primzahlen in der Faktorbasis näherungsweise halbiert. Dadurch entfallen im Siebschritt unnötige Testdivisionen und die Faktorisierung wird beschleunigt.

Die Faktorbasis für $n$ mit Schranke $B$ ist damit

$$F = \left\{ p \in \mathbb{P} \mid p \leq B,\ \left(\frac{n}{p}\right) = 1 \right\}.$$

### 2.4.2 Sieben der Kongruenzen

Nach Aufstellen der Faktorbasis kann das eigentliche Sieben der Kongruenzen ab der Wurzel von $n$ beginnen. Dabei wird $\sqrt{n}$ als Startwert benutzt, da dann die zu faktorisierenden Kongruenzen

$$(a_i^2 \bmod n) = a_i^2 - n$$

noch relativ klein sind und sich mit hoher Wahrscheinlichkeit über $F$ faktorisieren lassen. In der Praxis zeigt sich, dass ab der Wurzel von $n$ bereits bis

$$a_i < \sqrt{2n}$$

genügend $B$-glatte Kongruenzen gefunden werden. Dies bedeutet, dass im gesamten Siebprozess

$$a_i^2 < 2n$$

gilt und somit niemals echt modulo $n$ gerechnet, sondern stets nur $n$ subtrahiert werden muss. Im Folgenden sei daher

$$f(x) = x^2 - n$$

das Polynom, für das $B$-glatte Werte gesiebt werden.

Eine weitere Beobachtung ist, dass ab $\lceil \sqrt{n} \rceil$ die zu faktorisierenden Kongruenzen betragsmäßig größer werden und es daher unwahrscheinlicher ist, dass diese nur Primfaktoren aus $F$ besitzen. Daher erweitert man das Siebintervall,

sodass nicht ab der Wurzel von $n$ aufsteigend, sondern zentriert um $\lceil \sqrt{n} \rceil$ gesiebt wird. Dazu definiert man

$$Q(x) = \left( \lceil \sqrt{n} \rceil + x \right)^2 - n$$

und siebt nun $Q(0), Q(1), Q(-1), \ldots$ bis genügend Kongruenzen gefunden wurden. Der Nachteil dieser Methode ist, dass nun einige Kongruenzen negativ sein können.

Im Beispiel aus 2.2 ist für $n = 3007$, $\lceil \sqrt{3007} \rceil = 55$

$$Q(-2) \equiv 53^2 \equiv -2 * 3^2 * 11 \equiv (-1)^1 * 2^1 * 3^2 * 11^1 \pmod{3007}$$

eine Kongruenz, die auch innerhalb der Faktorbasis aus 2.2 zerfällt. Daher muss $-1$ zusätzlich zur Faktorbasis hinzugenommen werden, um mit dem Exponenten 0 eine positive und mit 1 eine negative Kongruenz zu markieren. Alle weiteren Schritte incl. des Matrixschrittes verlaufen dann analog. Somit enthält die Faktorbasis

$$F = \{-1, p_1, \ldots, p_k\}$$

nun ein Element mehr, wodurch auch die Matrix um eine Zeile erweitert wird und daher nun $k+2$ $B$-glatte Kongruenzen gesiebt werden müssen. Dies bedeutet allerdings nur einen geringen zusätzlichen Aufwand im Sieb- und Matrixschritt gegenüber dem Vorteil, dass nun mehr $B$-glatte Kongruenzen gefunden werden (siehe Kapitel 4) und somit auch schneller eine Faktorisierung von $n$ erzielt wird, sodass es sehr sinnvoll ist, auch negative Kongruenzen zuzulassen.

Wird nun ein Intervall der Länge $2L$,

$$I = [-L, +L],$$

mit Hilfe von $Q(x)$ gesiebt, so ist es nicht nötig, für alle $a_i \in I$ naiv $Q(a_i)$ auf Teilbarkeit durch alle Primzahlen $p$ aus der Faktorbasis $F$ zu prüfen, da die Kongruenzen nur in Abständen durch $p$ teilbar sind. Gilt nämlich für eine Primzahl $p \in F$, dass

$$Q(x) \equiv 0 \pmod{p}$$

für ein $x \in I$, so auch

$$\begin{aligned} Q(x+p) &\equiv \left( \lceil \sqrt{n} \rceil + x + p \right)^2 - n \\ &\equiv \left( \lceil \sqrt{n} \rceil + x \right)^2 + 2 \left( \lceil \sqrt{n} \rceil + x \right) p + p^2 - n \\ &\equiv Q(x) \\ &\equiv 0 \pmod{p}, \end{aligned}$$

sodass nach einer Kongruenz, die durch eine Primzahl aus der Faktorbasis teilbar ist, direkt alle weiteren im Intervall angegeben werden können.

Pro Intervall $I$ muss dazu allerdings zuvor noch pro Primzahl $p$ der Faktorbasis das erste $a_{1p} \in I$ ermittelt werden, für das $Q(a_{1p})$ durch $p$ teilbar ist. Dazu wird vor dem Siebschritt

$$x_{1p}^2 \equiv n \pmod{p}$$

für alle $p \in F$ per Algorithmus von Shanks-Tonelli gelöst und anschließend das kleinste $a_{1p} \in I$ ermittelt, das

$$\lceil \sqrt{n} \rceil + a_{1p} \equiv x_{1p} \pmod{p}$$

erfüllt. Nach Wahl von $a_{1p}$ ist gerade

$$Q(a_{1p}) \equiv \left(\lceil \sqrt{n} \rceil + a_{1p}\right)^2 - n \equiv x_{1p}^2 - n \equiv n - n \equiv 0 \pmod{p},$$

sodass für alle weiteren

$$a_{1i} = a_{1p} + (i-1)p,\ i \in \mathbb{N}$$

im Intervall der Wert $Q(a_{1i})$ durch $p$ teilbar ist. Zu beachten ist, dass man zwei Restklassen erhält, $x_{1p}$ und $x_{2p} = p - x_{1p}$, mit denen gesiebt werden muss.

### 2.4.3 Praktische Umsetzung des Siebschrittes

In der praktischen Umsetzung wird zunächst die Faktorbasis wie unter 2.4.1 beschrieben initialisiert. Das Siebintervall sei $I = [-L, +L]$. Zudem müssen für alle Primzahlen $p$ der Faktorbasis die Lösungen $x_{1p}$ (und $x_{2p} = p - x_{1p}$) von

$$x_{1p}^2 \equiv n \pmod{p}$$

und daraus die ersten $a_{1p}, a_{2p} \in I$ mit

$$\lceil \sqrt{n} \rceil + a_{1p} \equiv x_{1p} \pmod{p},$$

$$\lceil \sqrt{n} \rceil + a_{2p} \equiv x_{2p} \pmod{p}$$

berechnet werden. Beide Schritte müssen nur einmalig durchgeführt werden. Anschließend wird für das Siebintervall $I$ ein Array der Länge $2L$ mit den Werten $Q(x)$, $x \in I$ gefüllt und für alle Primzahlen und alle Vielfachen

$$a_{1i} = a_{1p} + (i-1)p,\ i \in \mathbb{N},$$

$$a_{2i} = a_{2p} + (i-1)p,\ i \in \mathbb{N}$$

$p$ maximal aus den Arrayeinträgen von $Q(a_{1i})$ und $Q(a_{2i})$ dividiert. Nachdem dieser Vorgang für alle Primzahlen aus der Faktorbasis durchgeführt wurde, können in einem linearen Durchlauf durch das Array alle $B$-glatten Kongruenzen $x$ bestimmt werden, denn diese sind gerade die Werte, für die der Arrayeintrag von $Q(x)$ bis auf 1 dividiert wurde (oder alternativ: die $x$, für die die getroffenen Primzahlen ab 1 aufmultipliziert wieder $Q(x)$ ergeben). Werden nun noch in einem Vektor pro $Q(x)$ die ausgeteilten Primfaktoren mit ihren Exponenten gespeichert, können alle faktorisierten Kongruenzen direkt für den Matrixschritt gespeichert werden.

Auch der Siebschritt kann nochmals beschleunigt werden. Da die $Q(x)$ für $x \in I$ bis in die Größenordnung von $n$ anwachsen können, sind für große $n$ besonders die Divisionen bzw. Multiplikationen aufwendig, die in der obigen Variante auftreten. Zudem werden nur wenige $Q(x)$ ganz über $F$ zerfallen und die restlichen am Ende sowieso verworfen, sodass die meisten Divisionen unnötig durchgeführt werden. Aus diesem Grund werden mit Hilfe des Logarithmus die Divisionen in Subtraktionen bzw. Multiplikationen in Additionen überführt, die auch für Zahlen mit mehreren hundert Stellen problemlos mit prozessornahen Integer-Typen schnell durchgeführt werden können.

Vor dem Sieben wird nun im Array der Länge $2L$ wie oben nicht $Q(x)$ sondern $\lceil \log(Q(x)) \rceil$ für alle $x \in I$ eingetragen, dies entspricht der Anzahl der Bits eines $Q(x)$. Das Array wird nun für alle Primzahlen $p$ der Faktorbasis ab den Startwerten $a_{1p}$ und $a_{2p}$ wie oben in Sprüngen im Abstand $p$ durchlaufen und statt $p$ dividiert jetzt $\lceil \log(p) \rceil$ subtrahiert (oder alternativ: auf die Arrayeinträge ab 0 $\lceil \log(p) \rceil$ addiert).

Der Nachteil der Variante ist allerdings, dass auf Grund der gerundeten Werte der Logarithmen nicht mehr definitiv entschieden werden kann, ob eine Kongruenz wirklich faktorisiert worden ist. Um am Schluss dennoch die faktorisierten Kongruenzen zu ermitteln, wird ein Schwellenwert $t$ („threshold“) festgesetzt, ab dem eine Kongruenz als faktorisiert angenommen wird. Zudem wird auf das Sieben mit Potenzen der Primzahlen verzichtet und statt dessen ein Beitrag im Schwellenwert $t$ berücksichtigt. In einem linearen Durchlauf durch das Array werden dann alle Einträge bestimmt, deren Werte nach den Subtraktionen kleiner oder gleich $t$ sind (bzw. alternativ um genau oder weniger als $t$ vom Wert $\lceil \log(Q(x)) \rceil$ abweichen), denn diese zerfallen sehr wahrscheinlich über der Faktorbasis. Alle so gefundenen Kongruenzen müssen per Testdivision anschließend nochmals über der Faktorbasis $F$ faktorisiert werden, um zu testen, ob diese wirklich über $F$ zerfallen und um deren exakte Faktorisierung für den Matrixschritt zu speichern.

Durch diesen Ansatz kann der benötigte Speicherplatz verringert und die Siebzeit stark verkürzt werden (siehe Kapitel 4). Zudem können wie zuvor die Primzahlen, von denen ein Arrayeintrag während der Subtraktionen getroffen wurde, in Listen gespeichert werden, um bei einer anschließenden Testdivision die Primzahlen aus der Faktorbasis einschränken zu können. Als heuristischer Wert hat sich nach [1] $t = 20$ bewährt.

### 2.4.4 Kombination zu einer Darstellung $a^2 \equiv b^2 \pmod{n}$

Wurden im Siebschritt genügend $B$-glatte Kongruenzen gefunden, so können alle Exponenten modulo 2 wie unter 2.3.2 beschrieben in eine Matrix eingetragen und der Kern dieser Matrix bestimmt werden. Die Kernvektoren geben anschließend die Kongruenzen $i \in I$ (Indexmenge $I$) an, die kombiniert werden müssen, um eine Quadratdarstellung

$$a^2 := \left( \prod_{i \in I} a_i \right)^2 \equiv \left( p_1^{\frac{1}{2}d_1} \cdots p_k^{\frac{1}{2}d_k} \right)^2 =: b^2 \pmod{n}$$

mit ausschließlich geraden Exponenten $d_i$ zu erhalten. Ein nicht trivialer Kern ist garantiert, falls mehr Kongruenzen als Primzahlen in der Faktorbasis (incl. $-1$) gefunden wurden, da die Matrix dann weniger Zeilen als Spalten besitzt. Da eine solche Kongruenz nicht notwendig eine Faktorisierung von $n$ liefert, sollte gleich eine Reihe von zusätzlichen Kongruenzen gefunden werden, bis der Matrixschritt gestartet wird. Enthält die Faktorbasis

$$F = \{-1, p_1, \ldots, p_k\}$$

$k+1$ Elemente und werden $k+1+M$ ($M \geq 1$) $B$-glatte Kongruenzen gefunden, so ist demnach ein $M$-dimensionaler Kern garantiert. Jeder dieser $M$ Kernvektoren liefert eine Quadratdarstellung von $n$. Die Wahrscheinlichkeit, dass eine

solche Quadratdarstellung nicht trivial ist und damit eine Faktorisierung von $n$ ermöglicht, ist $\geq 0,5$.

Dies ergibt sich aus der folgenden Überlegung: Jede gefundene Darstellung $a^2 \equiv b^2 \pmod{n}$ kann durch zweimalige Multiplikation mit der Inverse von $b \pmod{n}$ (falls diese existiert, ansonsten ist direkt ein Faktor gefunden), auf die Form

$$c^2 \equiv 1 \pmod{n} \tag{3}$$

gebracht werden. Auch eine solche Darstellung kann zur Faktorisierung von $n$ verwendet werden, da aus

$$a^2 \equiv b^2 \pmod{n},$$

$$a \not\equiv b \pmod{n}$$

auch

$$c \not\equiv \pm 1 \pmod{n}$$

folgt, sodass $gcd(n, c+1)$ und $gcd(n, c-1)$ eine nicht triviale Faktorisierung von $n$ liefern. Besteht $n$ aus $r$ verschiedenen Primfaktoren,

$$n = p_1 \cdots p_r,$$

so kann die Berechnung einer Quadratwurzel $c$ modulo $n$ auf die Berechnung der Quadratwurzeln modulo der $p_i$ zurückgeführt und anschließend eine Lösung modulo $n$ per Chinesischem Restsatz bestimmt werden. In $\mathbb{F}_{p_i}$ besitzt Gleichung (3) genau zwei Lösungen. Jede der $2^r$ Kombinationen aller Lösungen modulo $p_i$ kann per Chinesischem Restsatz zusammengefasst werden. Modulo $n$ sind aber bereits die zwei trivialen Lösungen 1 und $-1$ bekannt, alle weiteren Lösungen ermöglichen eine Faktorisierung von $n$. Die Wahrscheinlichkeit, eine der trivialen Lösung zu erhalten, ist damit

$$P_t = \frac{2}{2^r} \leq 0.5$$

für $r \geq 2$.

Wird daher der Siebschritt solange durchgeführt, bis wie oben beschrieben $k + 1 + M$ $B$-glatte Kongruenzen gefunden wurden und damit ein Kern der Dimension $M$ garantiert ist, so ist die Wahrscheinlichkeit, dass mindestens eine der pro Kernvektor erzielten Quadratdarstellungen von $n$ eine Faktorisierung liefert nach Gegenwahrscheinlichkeit

$$P_F = 1 - {P_t}^M \geq 1 - \frac{1}{2^M}.$$

Enthält die Faktorbasis somit $k + 1$ Elemente und werden $M = 10$ zusätzliche Kongruenzen vor dem Matrixschritt gesiebt, so wird mit einer Wahrscheinlichkeit größer $\frac{1023}{1024} \approx 0.999$ eine Faktorisierung gefunden.

### 2.4.5 Ermittlung der Faktoren per ggT

Ist der Kern der Exponentenmatrix berechnet, werden alle Kernvektoren durchlaufen und jeweils alle darin mit einer 1 markierten Kongruenzen $a_i$, $i \in I$, faktorisiert als

$$a_1^2 \equiv p_1^{e_{1,1}} \cdots p_k^{e_{k,1}} \pmod{n}$$
$$\vdots$$
$$a_r^2 \equiv p_1^{e_{1,r}} \cdots p_k^{e_{k,r}} \pmod{n}$$

mit Vielfachheit 1 aufmultipliziert:

$$a := \prod_{i \in I} a_i.$$

Dabei kann direkt modulo $n$ reduziert werden. Gesucht ist nach 2.3.2 eine Darstellung

$$\left(\prod_{i \in I} a_i\right)^2 \equiv \left(p_1^{\frac{1}{2}d_1} \cdots p_k^{\frac{1}{2}d_k}\right)^2 \pmod{n}$$

mit geraden Exponenten $d_j = \sum_{i \in I} e_{j,i}$ nach Konstruktion über die Exponentenmatrix in $\mathbb{F}_2$. Da die Summen $d_j$ der Exponenten der Primfaktoren am Schluss halbiert werden müssen, um eine Quadratdarstellung zu erhalten, können die Faktoren $p_i^{e_{i,j}}$ nicht direkt modulo $n$ aufmultipliziert werden, da im Voraus nicht bekannt ist, wann die Hälfte eines Exponenten erreicht ist. Statt dessen wird pro Kernvektor ein Exponentenvektor $d = (d_1, \ldots, d_k)$ angelegt, in dem die Exponenten der Primzahlen in den faktorisierten Kongruenzen aufaddiert werden. Sind alle Kongruenzen durchlaufen wird jeder Eintrag im Exponentenvektor halbiert und per schnellem Potenzieren

$$b := p_1^{\frac{1}{2}d_1} \cdots p_k^{\frac{1}{2}d_k} \pmod{n}$$

berechnet.

Mit $gcd(n, a + b)$ und $gcd(n, |a - b|)$ kann anschließend versucht werden, eine Faktorisierung von $n$ zu finden. Liefern alle Kernvektoren keinen Faktor, so muss zum Siebschritt zurückgekehrt und weitere $B$-glatte Kongruenzen gefunden werden. Vor einer erneuten Kernberechnung der Exponentenmatrix sollte allerdings nicht eine, sondern wieder eine Reihe von neuen faktorisierten Kongruenzen gefunden werden, um die Wahrscheinlichkeit zu erhöhen, im nächsten Durchlauf eine Faktorisierung zu finden. Obwohl die zuvor gefundenen Kongruenzen keine Faktorisierung ermöglichten, sollten alle Kongruenzen auch im nächsten Durchlauf wieder in die Matrix eingetragen werden, da eine der neu faktorisierten Kongruenzen unter Umständen mit einer der vorherigen zu einer Quadratdarstellung von $n$ kombiniert werden kann.

Um eine Faktorisierung zu finden muss zudem nur eine Basis des Kerns und keine Linearkombinationen aus Basisvektoren untersucht werden. Liefern nämlich alle Basisvektoren nur $a$ und $b$ wie in 2.4.4 mit trivialen $gcd$ oder äquivalent

$$a^2 \equiv b^2 \pmod{n},$$
$$a \equiv \pm b \pmod{n},$$

so gilt auch für die Summe der Kernvektoren zweier trivialer Darstellungen

$$a_1 \equiv \pm b_1 \pmod{n},$$
$$a_2 \equiv \pm b_2 \pmod{n},$$

dass

$$a_{12} := a_1 * a_2 \equiv \pm b_1 * b_2 =: b_{12} \pmod{n}$$

und somit keine Faktorisierung mit dieser Linearkombination gefunden wird.

## 2.5 Large Prime Variation

An der in den vorherigen Abschnitten eingeführten Standardvariante des Quadratischen Siebes können weitere Modifikationen vorgenommen werden, um den Faktorisierungaufwand zu minimieren. Im Siebschritt werden die für den Matrixschritt nötigen $B$-glatten Kongruenzen gefunden, jedoch werden dazu alle Kongruenzen nur einmal bzw. immer nur in Intervallen frei wählbarer Länge betrachtet. Im Matrixschritt jedoch müssen alle Kongruenzen zusammen ausgewertet werden, sodass gerade dieser Schritt wegen des nötigen Speicherplatzes und Rechenaufwandes eine Einschränkung des Verfahrens darstellt. Da die Matrixgröße nach 2.3.2 direkt mit der verwendeten *Smoothness Bound B* und daher mit der Anzahl benötigter Kongruenzen zusammenhängt, wirkt sich eine eingeschränkte Matrixgröße auch auf den Siebschritt aus, da bei kleiner Schranke $B$ auch weniger $B$-glatte Kongruenzen gefunden werden.

Die „Large Prime Variation“ bietet eine einfache Möglichkeit, bei unveränderter *Smoothness Bound* mehr Kongruenzen für den Matrixschritt zu sammeln, indem „fast“ $B$-glatte Kongruenzen gesucht werden. Dazu werden alle Kongruenzen, die nicht $B$-glatt sind und nur deshalb nicht innerhalb der Faktorbasis zerfielen, weil ein einziger Primfaktor größer $B$ nach der Faktorisierung übrig blieb, in einem Zwischenschritt nach dem Sieben und vor dem Lösen der Matrix gesondert behandelt. Die Idee ist, dass zwei solcher Kongruenzen $a_1$ und $a_2$ mit gleichem großen Primfaktor $(P)$

$$a_1^2 - n = p_1^{e_{1,1}} \cdots p_k^{e_{k,1}} * P = y_1 * P,$$

$$a_2^2 - n = p_1^{e_{1,2}} \cdots p_k^{e_{k,2}} * P = y_2 * P,$$

zwar für sich nicht im Matrixschritt verwendet werden können, deren Produkt

$$(a_1 * a_2)^2 \equiv y_1 * y_2 * P^2 \pmod{n}$$

den Primfaktor $P$ jedoch mit Exponent 2 enthält, sodass der Exponent von $P$, der „außerhalb“ der Matrix eingetragen werden müsste, in $\mathbb{F}_2$ nicht vorkommt. Durch die Kernberechnung können die addierten Exponenten der Primzahlen aus der Faktorbasis von

$$(a_1 * a_2)^2 \equiv p_1^{e_{1,1}+e_{1,2}} \cdots p_k^{e_{k,1}+e_{k,2}} * P^2 \pmod{n} \tag{4}$$

zu geraden Exponenten kombiniert werden, sodass eine neue Kongruenz der „Large Prime Variation“ ohne den Exponenten 0 von $P$ wie jede andere gesiebte Kongruenz in die Matrix eingetragen wird.

Analog werden $r$ Kongruenzen mit gleichem großen Primfaktor $P$ zu $r-1$ neuen Kongruenzen

$$(a_1 * a_i)^2 \equiv y_1 * y_i * P^2 \pmod{n} \tag{5}$$

für $i \in \{2, \ldots, r\}$ kombiniert. Da in allen Kongruenzen der „Large Prime Variation“ der Primfaktor $P$ wegen des geraden Exponenten nicht in die Matrix eingetragen werden muss, bleibt die Zeilenanzahl der Matrix konstant, während die zusätzlichen Kongruenzen den Siebschritt beschleunigen.

Zu beachten ist lediglich, dass zur Ermittlung der Faktoren in 2.4.5 für eine per „Large Prime Variation“ kombinierte Kongruenz zusätzlich zu den aufmultiplizierten $a_i$ in $a$ und den aufaddierten Exponenten der Primzahlen aus der

Faktorbasis in einer zusätzlichen Variable $c$ die großen Primfaktoren $P_i$, die zu diesen Kongruenzen gehören, mit Vielfachheit 1 aufmultipliziert werden müssen. Dies bedeutet, dass analog 2.4.5

$$a := \prod_{i \in I} a_i \pmod{n},$$

$$b := p_1^{\frac{1}{2}d_1} \cdots p_k^{\frac{1}{2}d_k} \pmod{n},$$

noch

$$c := \prod_{i \in I} P_i \pmod{n}$$

berechnet werden muss. Dann gilt

$$a^2 \equiv (b * c)^2 \pmod{n}$$

und die Faktorisierung kann per $gcd(n, a + bc)$ und $gcd(n, a - bc)$ ermittelt werden.

Im Beispiel aus 2.2 für $n = 3007$, $\lceil \sqrt{3007} \rceil = 55$ erhält man mit

$$Q(-3) \equiv 52^2 \equiv -3 * 101 \pmod{3007},$$

$$Q(-6) \equiv 49^2 \equiv -2 * 3 * 101 \pmod{3007}$$

zwei Kongruenzen, die beide den relativ großen Primfaktor 101 enthalten und deshalb nicht als $B$-glatt identifiziert werden. In der „Large Prime Variation“ werden beide Kongruenzen neu zu

$$(52 * 49)^2 \equiv 2 * 3^2 * 101^2 \pmod{3007}$$

kombiniert und können so ohne $101^2$ in die Matrix eingetragen werden.

Um effizient Paare von Kongruenzen mit gleichem Primfaktor zu finden, wird der Siebschritt so modifiziert, dass alle nicht vollständig faktorisierten Kongruenzen

$$a_i^2 - n = p_1^{e_{1,i}} \cdots p_k^{e_{k,i}} * P_i$$

mit großem Primfaktor $P_i$ in einer Liste als $(a_i, P_i)$ abgespeichert werden. Zur Auswertung wird die Liste nach dem zweiten Eintrag sortiert. In einem linearen Durchlauf können dann alle Kongruenzen einer Sequenz des gleichen großen Primfaktors direkt zu neuen Kongruenzen kombiniert werden, indem diese wie in (5) mit dem ersten Vorkommen in der Sequenz multipliziert werden. Wird zusätzlich die Faktorisierung aller Kongruenzen über der Faktorbasis in der Liste gespeichert, so können die Exponenten wie in (4) direkt zur Faktorisierung des Produktes addiert werden.

Des Weiteren ist es nicht effizient, alle unvollständig faktorisierten Kongruenzen zur „Large Prime Variation“ abzuspeichern, da es bei größeren verbleibenden Primfaktoren auch zunehmend unwahrscheinlicher wird, ein zweites Vorkommen zur Kombination zu finden. Nach [1] hat es sich bewährt, alle Kongruenzen mit einem verbleibenden Primfaktor aus $[B, 100B]$ zur Endauswertung beizubehalten und alle mit noch größerem Primfaktor direkt zu verwerfen. Zudem ist die Technik nicht auf verbleibende Primfaktoren in den Faktorisierungen beschränkt. Jeder beliebige Rest kann verwendet werden, sofern dieser

durch mindestens ein zweites Vorkommen zu einem Quadrat kombiniert wird. Interessant ist zudem, wie wahrscheinlich es ist, überhaupt weitere Vorkommen für große Primfaktoren zu finden. Die Antwort gibt das Geburtstagsparadoxon, das besagt, dass die Wahrscheinlichkeit für doppelte Vorkommen am Anfang eines Siebschrittes noch sehr gering ist, während nach genügend großer Anzahl an gesiebten Kongruenzen bis zur Auswertung der Liste in der „Large Prime Variation“ mehrfache Vorkommen häufiger auftreten werden. Nach [2] kann mit dieser Methode die Siebzeit für große Zahlen bis auf ein Sechstel reduziert werden. Nachteil der Methode ist, dass durch Verschmelzen von jeweils zwei Faktorisierungen die Anzahl der Primzahlen pro Kongruenz zunimmt, wodurch in die Matrix auch mehr Einsen eingetragen werden. Dies kann der in 2.3.2 erwähnten effizienten Speicherung der Matrizen und der Anwendung schneller Matrixalgorithmen entgegenwirken.

## 2.6 Computeranwendung und Parallelisierung

Das Quadratische Sieb beinhaltet eine Vielzahl an Techniken, die besonders durch Computereinsatz elegant gelöst werden können.

Da die Matrix über $\mathbb{F}_2$ erstellt wird, kann diese binär in der verfügbaren Maschinenwortlänge dargestellt werden. Dazu wird die Matrix als zweidimensionales Integer-Array erstellt. Ein typischer Integer beinhaltet 32 Bit, sodass die Matrix als Array mit

$$\begin{aligned} \text{Zeilenanzahl} &= \#\text{Faktorbasis}, \\ \text{Spaltenanzahl} &= \left\lceil \frac{\#\text{Kongruenzen}}{32} \right\rceil \end{aligned}$$

initialisiert wird. Die Matrixeinträge müssen entsprechend mit „shift left (shl)“- und „shift right (shr)“-Operationen beschrieben werden. Zeilenadditionen im Gauß-Algorithmus werden zu „xor“-Operationen der einzelnen Integer pro Zeile. Alle diese Operationen sind prozessornah und werden schnell ausgeführt.

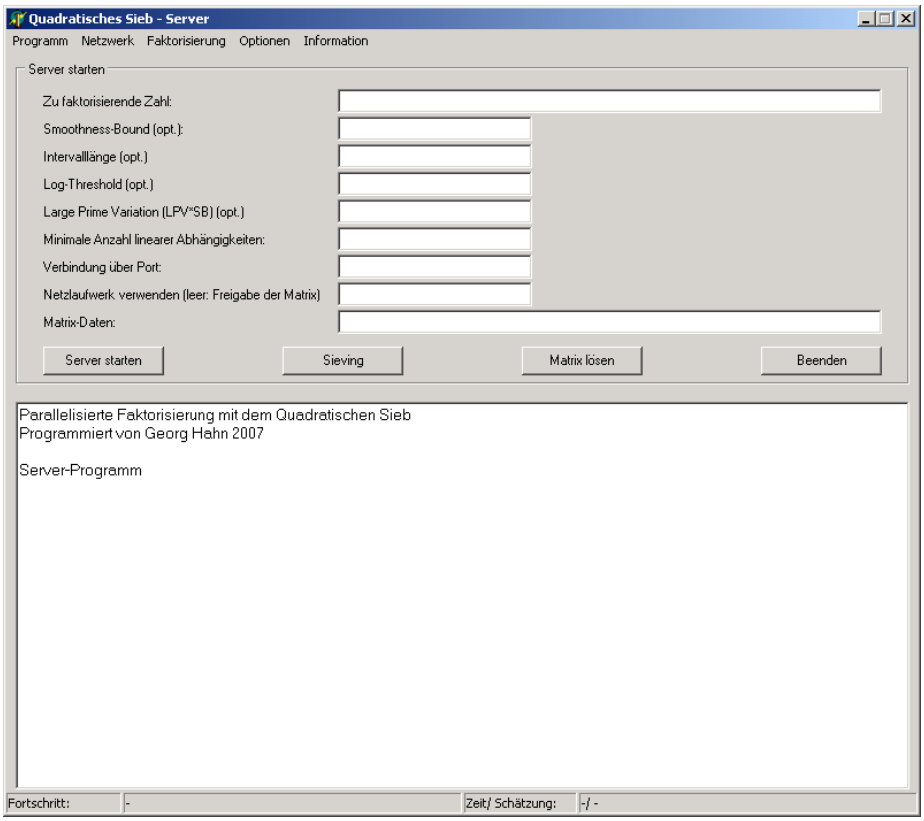
Zudem ist wie bereits in 2.4.3 erwähnt der Siebschritt mit Logarithmen schnell ausführbar. Per Logarithmus sind die Divisionen in Subtraktionen bzw. Multiplikationen in Additionen überführbar und können so auch für Zahlen mit mehreren hundert Stellen problemlos mit prozessornahen Integer-Typen schnell durchgeführt werden. Die Berechnung des Logarithmus entfällt, da es reicht, die zur Repräsentation der Zahlen nötigen Bits als gerundete Logarithmen zu verwenden.

Hauptvorteil des Quadratischen Siebes ist jedoch die Parallelisierung. Die im Siebschritt durchsuchten Intervalle können völlig unabhängig voneinander bearbeitet werden, sodass der Siebschritt auf beliebig viele Computer verteilbar ist. Wird der Siebschritt auf $s$ Clients verteilt, so wird das Sieben auch $s$ mal schneller als mit einem Client abgeschlossen. Dazu muss allen Clients die zu faktorisierende Zahl und die sonstigen notwendigen Parameter mitgeteilt werden. Jeder Client bekommt einen Siebbereich, der anhand der $L$-Heuristik aus 2.4.1 als $L/\#\text{Clients}$ festgelegt werden kann. Alle gefundenen Kongruenzen werden an den Server gesendet, der diese in die Matrix einträgt. Die Kernberechnung der Matrix muss schließlich von einem Computer alleine durchgeführt werden.

# 3 Realisierung eines Netzwerks zur parallelisierten Faktorisierung

Im Rahmen der Arbeit wurde das Quadratische Sieb als Server-Client-Netzwerk realisiert. Als Programmiersprachen dienten Java und Borland Delphi. Die ersten Versionen des Programms wurden in Java geschrieben. Als sich jedoch herausstellte, dass die Anwendung zu langsam sein würde, wurde das gesamte Programm auf eine kompilierte Version umgeschrieben. Im Folgenden soll auf Besonderheiten des Programms eingegangen werden.

## 3.1 Das Serverprogramm




Der Server unterstützt alle in der Arbeit vorgestellten Faktorisierungmethoden des Quadratischen Siebes. In einer Eingabemaske werden die zu faktorisierende Zahl und sämtliche folgenden Parameter angegeben:

- Smoothness Bound $B$
- Siebintervalllänge $L$
- Threshold für das Sieben der Intervalle mit Logarithmen
- Parameter für die „Large Prime Variation“
- Gewünschte minimale Anzahl an linearen Abhängigkeiten

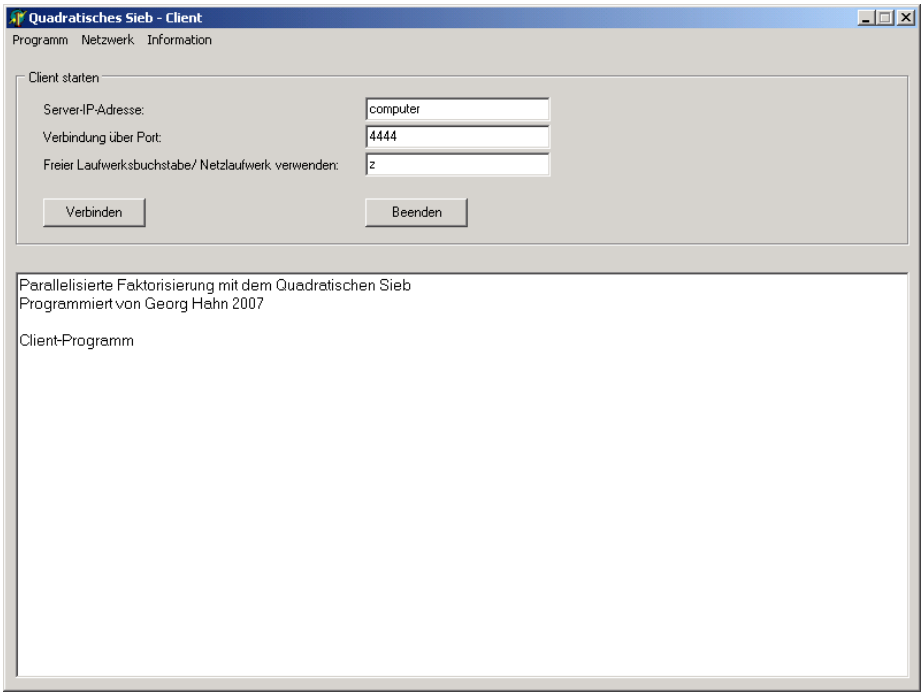
- Siebmethode: Trial Division, Intervalle, Intervalle logarithmisch
- Legendre-Symbol testen
- Symmetrisch um die Wurzel sieben
- Large Prime Variation aktivieren/ deaktivieren

Von diesen Parametern müssen nur die zu faktorisierende Zahl und die minimale Anzahl an linearen Abhängigkeiten angegeben werden, alle anderen Parameter werden vom Programm übernommen, falls diese angegeben sind und ansonsten automatisch per Heuristiken bestimmt. Pro Zahl wird ein Verzeichnis für die Matrixdaten angegeben, in das alle Dateien mit den faktorisierten Kongruenzen abgelegt werden. Der Server gibt wahlweise das Verzeichnis im Netzwerk frei, damit die Clients die Siebdateien direkt darin ablegen können oder tauscht die Dateien über ein Drittlaufwerk im Netzwerk aus, auf das Server und Clients zugreifen können. Für die Eingabezahl kann per Menüpunkt „Faktorisierung → Information“ eine Übersicht über alle heuristischen Werte sowie die Größe der Faktorbasis erstellt werden, um diese ggf. durch eine manuelle Angabe zu korrigieren oder über die Faktorbasis den Aufwand im Matrixschritt abschätzen zu können.

Zu Beginn des Siebprozesses wird ein Port gewählt und per Button „Server starten“ geöffnet. Alle Clients können sich anschließend über diesen Port mit dem Server verbinden. Eine Statusanzeige gibt einen Überblick über alle zur Zeit gefundenen Clients mit Clientnummer, Computername und IP-Adresse. Mit „Sieving“ wird der Siebprozess gestartet. Dieser kann jederzeit abgebrochen werden, ein Client kann während des Siebens aus dem Netzwerk austreten oder neue Clients können im Netzwerk angemeldet werden. Das Serverprogramm überwacht ständig das Netzwerk und verteilt automatisch neue Bereiche an alle Clients. Ein abgebrochener Siebvorgang kann jederzeit mit einer neuen Anzahl an Clients und neu gewählten Parametern in dem zuletzt gesiebten Intervall fortgesetzt werden.

Im Textfeld auf der Oberfläche schreibt der Server ein Protokoll aller ausgeführten Aktionen mit. In der unteren Programmleiste wird zudem ständig die bisher gefundene und die benötigte Anzahl an Kongruenzen ausgegeben, die verstrichene Zeit angezeigt und eine Schätzung der noch benötigten Zeit ausgegeben.

## 3.2 Der Siebschritt




Das Sieben der Kongruenzen wird ausschließlich von den Clients übernommen. Dazu wird in der Eingabemaske des Clients die Serveradresse und der geöffnete Port angegeben. Zusätzlich muss ein freier Laufwerksbuchstabe angegeben werden, unter dem der Client die Siebdateien mit dem Server austauscht. Nach „Verbinden“ geht der Client in eine Dauerverbindung zum Server über, sodass anschließend alle Aktionen vom Server aus gesteuert werden können, bis sich der Client wieder aus dem Netzwerk abmeldet. Das Protokoll, das von Server zu Client verwendet wird, lautet folgendermaßen:

1. „I *Zahl SB Legendre Siebmethode Symmetrisch_sieben LPV Laufwerk*“
2. „S *untererBereich obererBereich*“
3. „A“

Unter 1. initialisiert der Client eine neue Faktorbasis zur Faktorisierung der angegebenen Zahl mit Primzahlen bis zur „Smoothness Bound“ *SB*. Dabei gibt $Legendre \in \{0,1\}$ an, ob zur Auswahl der Primzahlen das Legendre-Symbol berechnet werden soll. Als Siebmethode wird „t“ (Trial-Division), „i“ (Intervalle) oder „lX“ (Intervalle mit Logarithmen und Threshold X) gesendet. Die Werte $Symmetrisch\_sieben \in \{0,1\}$ und $LPV \in \mathbb{N}$ geben an, ob strikt über oder zentiert um die Wurzel gesiebt werden soll und ob die „Large Prime Variation“ verwendet wird, wobei der Parameter auch direkt das Annahmeintervall $[B, LPV * B]$ für große Primzahlen angibt ($LPV = 0$ falls Option deaktiviert). Der Parameter *Laufwerk* gibt an, ob das Matrixverzeichnis freigeben wurde.

Zur Initalisierung der Faktorbasis wird das Sieb des Eratosthenes verwendet. Alle Rechenoperationen der Langarithmetik werden mit der schnellen „NX Numerics Library“ [6] von M. Martin durchgeführt. Da während des Siebprozesses

viele Daten gespeichert und schnell abrufbar sein müssen, werden stets Arrays mit Verdoppelungsstrategie verwendet.

Unter 2. bekommt der Client die Anweisung, das Intervall

$$\left[\lceil\sqrt{n}\,\rceil + untererBereich, \lceil\sqrt{n}\,\rceil + obererBereich\right]$$

zu sieben. Dazu wird die in der Initialisierung gesendete Siebmethode verwendet. Wird symmetrisch gesiebt, bearbeitet der Client anschließend noch das Intervall

$$\left[\lceil\sqrt{n}\,\rceil - obererBereich, \lceil\sqrt{n}\,\rceil - untererBereich\right]$$

unter der Wurzel. Da die Intervalle vom Server nicht eingeschränkt werden, teilt der Client ein zu großes Siebintervall automatisch in Intervalle zur Länge $l = 100000$ ein und siebt diese nacheinander, um den benötigten Arbeitsspeicher gering zu halten. Alle Kongruenzen werden nacheinander berechnet und per Probedivision über der Faktorbasis zerlegt, wobei direkt ein String der Form „$a_i$ $(-1)$ *Faktoren*“ mit der Faktorisierung mitgeschrieben wird. Dabei bezeichnet $a_i$ die aktuelle Position im Intervall, an der Stelle $(-1)$ steht 0 für eine positive und 1 für eine negative Kongruenz. Die Faktoren werden in der Darstellung *Primfaktor:Exponent* mit Leerzeichen getrennt angehängt, sodass nur die wirklich gefundenen Primfaktoren mit ihren Exponenten größer Null abgespeichert werden. Der Client öffnet eine Datei mit Namen des unteren Bereiches und schreibt darin die gefundenen Faktorisierungsstrings. Jede Datei bekommt zudem noch einen Header mit Informationen zu den Parametern, mit denen der Bereich gesiebt wurde: *Zahl SB Legendre Siebmethode uBereich oBereich Symmetrisch_sieben LPV*. In die letzte Zeile wird bei Abschluss des Siebintervalls noch die Anzahl der gefundenen und in der Datei abgelegten Kongruenzen geschrieben. Diese Daten werden im Matrixschritt benötigt. Zudem wird nach Abschluss eines Siebvorganges die Gesamtanzahl an gefundenen Kongruenzen pro Intervall an den Server gesendet, damit dieser rechtzeitig zum Matrixschritt wechseln kann.

In einem weiteren Array mit Verdoppelungsstrategie werden alle unvollständigen Kongruenzen für die „Large Prime Variation“ gespeichert, falls diese aktiviert ist, und die zwischengespeicherten Kongruenzen nach jedem komplett gesiebten Intervall ausgewertet. Zur Sortierung wird „Quicksort“ verwendet. In der Arbeit hat es sich gezeigt, dass es in der „Large Prime Variation“ zu lange dauert, die Faktorisierungsstrings zu speichern und zur Faktorisierung des Produktes zweier Kongruenzen zu „mischen“. Schneller ist es, die Faktorisierung des Produktes in einer speziellen Methode ein zweites Mal zu berechnen, die zwei Kongruenzen gleichzeitig faktorisiert und für beide einen einzigen Faktorisierungsstring ausgibt, der gerade der Faktorisierung des Produktes entspricht. Auf diese Weise wird auch der benötigte Speicherplatz verringert.

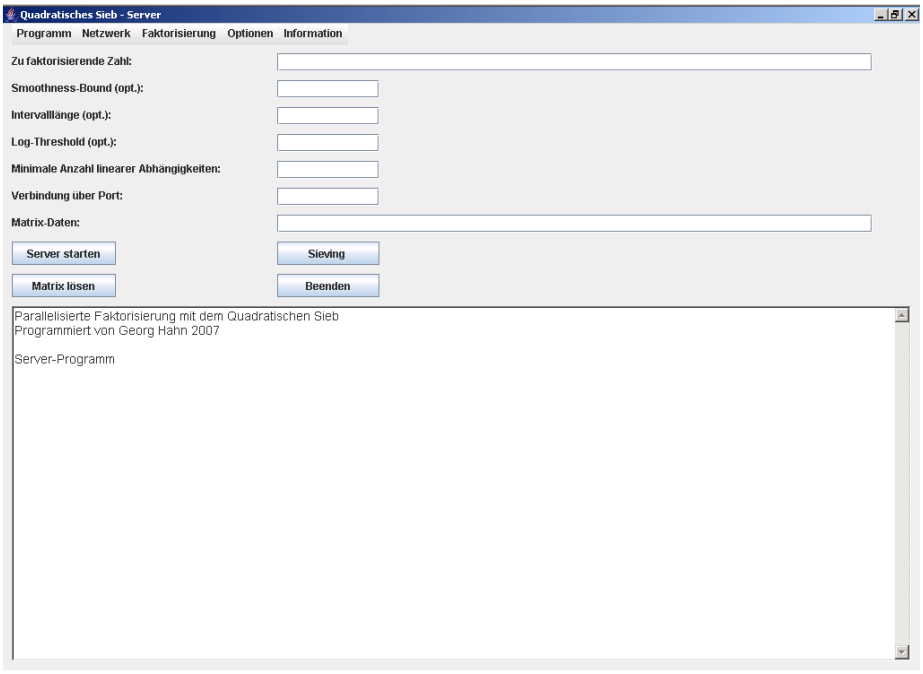
Der Client schreibt im Textfeld auf der Oberfläche ein Protokoll aller ausgeführten Aktionen mit. Ist eine Anweisung abgearbeitet, wird ein String „B“ (Bereit) an den Server gesendet, um eine neue Anweisung anzufordern. Der Client kann manuell abgebrochen werden, wobei ein aktueller Siebbereich zuvor noch abgeschlossen wird. Mit „A“ meldet sich der Client dann zum nächstmöglichen Zeitpunkt beim Server ab.

Unter 3. sendet der Server den Abbruchbefehl des Siebvorganges an alle Clients. Dies tritt bei manuellem Abbruch ein oder falls ausreichend Kongruenzen gefunden wurden. Der Client meldet sich daraufhin jedoch nicht ab, sondern bleibt in Verbindung zum Server, um mit einem neuen Initialisierungsbefehl den nächsten Faktorisierungsvorgang starten zu können.

## 3.3 Kernberechnung und Faktorisierung

Der Server aktualisiert während des gesamten Siebprozesses die von den Clients gesendete Gesamtanzahl an faktorisierten Kongruenzen. Sind ausreichend Kongruenzen gefunden, wird der Abbruchbefehl an alle Clients gesendet und der Matrixschritt gestartet. Dazu sucht das Programm automatisch nach allen Dateien mit Siebinformationen, die zur Eingabezahl gehören, wobei gleichzeitig analysiert wird, ob einige Bereiche mit unterschiedlichen Parametern, beispielsweise mit unterschiedlicher „Smoothness Bound“ bearbeitet wurden. Dies bedeutet nämlich, dass einige Bereiche mit verschieden großer Faktorbasis gesiebt wurden, wodurch auch die Zeilenanzahl der Exponentenmatrix für die einzelnen Siebbereiche variiert. Der Server erweitert automatisch die Matrix so, dass alle Dateien verwendet werden können und trägt die Kongruenzen modulo 2 ein. Dabei wird analog 2.6 die Integer-Wortlänge von 32 ausgenutzt und die Exponenten mit „shift-left“ und „shift-right“ in die Bits eingetragen. Die Zeilenaddition wird per bitweisem „xor“ der Integer, die zur Speicherung der Nullen und Einsen pro Zeile verwendet werden, schnell durchgeführt. Für die Kernberechnung wird der Gauß-Jordan-Algorithmus benutzt. Die Auswertung der Kernvektoren geschieht analog 2.4.5 und 2.5. Da in den Dateien alle Primfaktoren in der Notation *Primfaktor:Exponent* angegeben sind, in der Matrix jedoch eine Zeile für eine Primzahl aus der Faktorbasis steht, muss zum Eintragen eines Exponenten in die Matrix zuerst der zu einer Primzahl gehörige Index $i$ in der Faktorbasis ermittelt werden. Anschließend kann der Exponent modulo 2 in die Zeile $i$ und die Spalte der aktuellen Kongruenz eingetragen werden. In der Arbeit hat sich gezeigt, dass die Verwendung einer Binären Suche in der Faktorbasis die schnellsten Resultate liefert. Der Grund ist, dass für eine „sparse matrix“ so wenige Exponenten pro Spalte eingetragen werden müssen, dass es günstiger ist, den Index pro Exponent jedesmal neu zu suchen statt die Spalte linear zu durchlaufen. Der Server gibt am Ende die erste gefundene Kongruenz und den daraus resultierenden Faktor aus. Falls keine Faktorisierung berechnet werden konnte, wird eine Fehlermeldung ausgegeben. In diesem Fall sollte die minimale Anzahl an linearen Abhängigkeiten analog 2.4.5 erhöht und der Siebschritt fortgesetzt werden. Als Ausgabe können optional auch die gesamte binäre Matrix, die umgeformte Matrix nach Gauß-Jordan-Algorithmus, alle Basisvektoren des Kerns und alle daraus resultierenden Quadratdarstellungen angezeigt werden. Diese Option sollte jedoch nur für kleine Matrizen gewählt werden.

## 3.4 Die Java-Anwendung




Die erste Version des Programmes zur Arbeit als Server-Client-Netzwerk wurde in Java programmiert. Diese Anwendung arbeitet so wie die Windows-Version, ist allerdings auf Grund der Interpretierung statt Compilierung und der weniger optimierten „BigInteger“-Klasse deutlich langsamer. Außerdem wird die „Large Prime Variation“ nicht unterstützt.

# 4 Theoretische Laufzeit und Beispiele paralleler Faktorisierung

In diesem Kapitel werden einige Beispiele parallelisierter Faktorisierung vorgestellt und der Einfluss bestimmter Parameter verdeutlicht.

Der Siebschritt der folgenden Faktorisierungen wurde auf einem Client, Athlon XP 2600+ (1917 Mhz), 512 MB RAM, Betriebssystem Windows 2000 SP4, durchgeführt. Die verwendeten Zahlen $n$ in den Beispielen sind stets Produkt aus zwei Primfaktoren gleicher Stellenanzahl, wobei „Stellen“ die Dezimalstellenanzahl von $n$ angibt. Das Zeitformat ist immer in HH:MM:SS.MS angegeben.

## 4.1 Siebmethoden

Die Unterschiede zwischen den Siebmethoden „Trial Division“, „Intervalle“ und „Intervalle mit Logarithmen“ sind in der folgenden Tabelle zu sehen. Faktorisiert wurde $n = 563905175409432219211$ (21 Stellen).

| | |
|---|---|
| Trial Divison | 00:01:17.172 |
| Intervalle | 00:00:47.063 |
| Intervalle mit Logarithmen | 00:00:04.421 |
| Matrixschritt (für alle drei Methoden) | 00:00:00.062 |

Alle in der Arbeit beschriebenen Zusatzmethoden des Siebschrittes wurden deaktiviert. Die „Smoothness Bound“ wurde per Heuristik auf $B = 895$ gesetzt und eine optimale Intervalllänge von $L = 5200000$ manuell gewählt, in der gerade ausreichend Kongruenzen gefunden werden.

Die hohe Geschwindigkeit der Siebmethode „Intervalle mit Logarithmen“ wird für größere Zahlen deutlich sichtbar:

| Stellen | Intervalle | Intervalle mit Logarithmen |
|---|---|---|
| 21 | 00:00:47.063 | 00:00:04.421 |
| 31 | 00:16:14.125 | 00:01:21.532 |

Wie zuvor wurden alle Zusatzmethoden des Siebschrittes deaktiviert und manuell eine optimale Intervalllänge gewählt. Es ist deutlich zu sehen, dass die Siebmethode mit Logarithmen die weit besten Resultate liefert.

Nun soll die Auswirkung einzelner Parameter getestet werden. Alle folgenden Siebschritte wurden mit der Logarithmus-Variante durchgeführt. Exemplarische Faktorisierungen mit verschiedenen Optionen werden mit

$$\begin{aligned} n_{40} &= 4108131370631997507088207501257298124693 \\ &= 61510511726922465953 * 66787468601629502581 \end{aligned}$$

(40 Stellen) nach [4] durchgeführt.

## 4.2 Legendre-Symbol

Die Faktorbasis kann wie in 2.4.1 durch Berechnung des Legendre-Symbols für alle Primzahlen verkleinert werden. Die folgende Tabelle zeigt die Größe der Faktorbasis $\#F$ für verschiedene Zahlen:

| Stellen | $\#F$ ohne Legendre-Symbol | $\#F$ mit Legendre-Symbol |
|---|---|---|
| 20 | 119 | 52 |
| 30 | 647 | 340 |
| 40 | 2808 | 1388 |

Da ca. die Hälfte aller Elemente in $\mathbb{F}_p$ quadratische Reste sind, kann wie erwartet die Größe der Faktorbasis halbiert werden. Dadurch entfallen im Siebschritt unnötige Divisionen, wodurch die Faktorisierung von $n_{40}$ beschleunigt wird:

| ohne Legendre-Symbol | mit Legendre-Symbol |
|---|---|
| 00:21:57.097 | 00:01:00.672 |

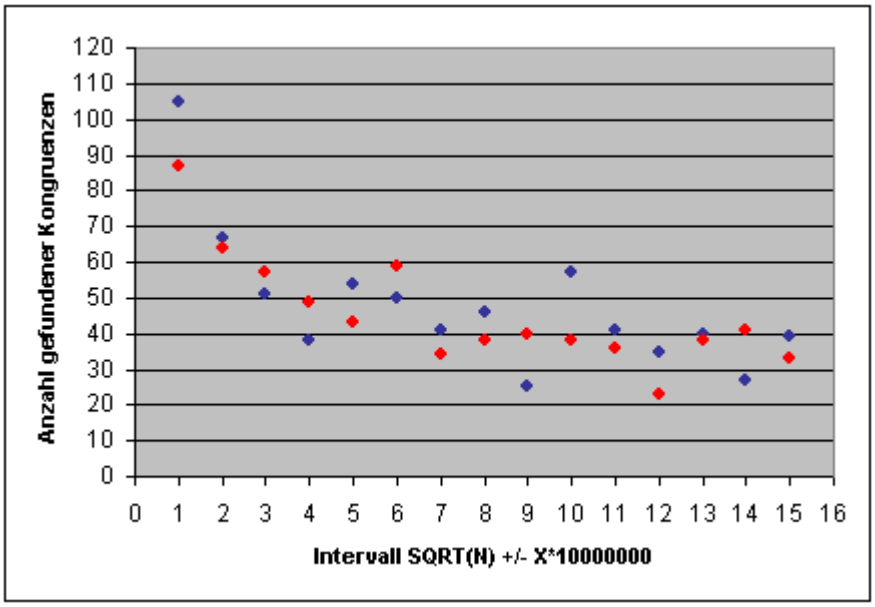
Die richtige Wahl der Faktorbasis spielt eine doppelte Rolle. Zum einen wird die Anzahl der Divisionen annähernd halbiert. Wichtiger ist jedoch, dass bei kleinerer Faktorbasis auch nur halb so viele $B$-glatte Kongruenzen gefunden werden müssen. Daher ist ein deutlicher Unterschied zwischen den Faktorisierungszeiten zu sehen.

## 4.3 Zentrierte Siebintervalle um die Wurzel

Im Kapitel 2.4.2 wurde bereits beschrieben, dass in den Siebbereichen, die weiter von der Wurzel von $n$ entfernt sind, auch weniger $B$-glatte Kongruenzen gefunden werden, da es für größere Werte von $Q(x)$ unwahrscheinlicher wird, über der festen Faktorbasis zu zerfallen. Die Faktorisierung von $n_{40}$ mit Legendre-Symbol und Siebbereichen zentriert um die Wurzel liefert:

| ohne symmetrische Intervalle | mit symmetrischen Intervallen |
|---|---|
| 00:01:00.672 | 00:00:52.172 |

Hier fällt der Zeitgewinn nicht so deutlich aus. Allerdings ist im folgenden Diagramm zu sehen, dass die Anzahl gefundener Kongruenzen für $n_{40}$ ab der Wurzel auf ca. ein Drittel abfällt, bis die Faktorisierung gefunden wird. Die blaue Datenreihe gibt die Anzahl in den Intervallen über der Wurzel wieder, die rote entsprechend unter der Wurzel. Es ist zu sehen, dass pro Intervall bis auf statistische Schwankungen etwa gleich viele Kongruenzen über und unter der Wurzel gefunden werden. Daher sollte das Sieben in zentrierten Intervallen um die Wurzel genutzt werden, um anfangs auch viele $B$-glatte Kongruenzen in den unter der Wurzel gelegenen Intervallen zu finden, da andernfalls die pro unterem Intervall nicht genutzten Kongruenzen erst nach Sieben mehrerer Intervalle über die Wurzel kompensiert werden können.




## 4.4 Large Prime Variation

Schließlich wird die „Large Prime Variation“ in der Praxis untersucht. Die folgenden zwei Tabellen zeigen die Faktorisierungszeiten von $n_{40}$ mit allen vorherigen Optionen sowie einer Variation mit unterschiedlichen Siebintervalllängen $L$ bzw. Annahmeintervallen $[B, V * B]$ für verbleibende Primfaktoren:

| Parameter | # partieller Kongruenzen | # kombinierter Kongruenzen | Zeit |
|---|---|---|---|
| $L = 10000000$ $V = 100$ | 14479 | 263 | 00:00:41.922 |
| $L = 50000000$ $V = 100$ | 8156 | 798 | 00:00:17.297 |
| $L = 10000000$ $V = 500$ | 19779 | 269 | 00:00:38.406 |
| $L = 50000000$ $V = 500$ | 11098 | 820 | 00:00:17.421 |

Die angegebene Anzahl partieller bzw. kombinierter Kongruenzen ist jeweils die Summe aller Kongruenzen, die bis zur Faktorisierung von $n_{40}$ ausgewertet wurden. Es ist zu sehen, dass bei gleichem Annahmeintervall $[B, V * B]$ die Anzahl der kombinierten Kongruenzen stark variiert. Der Grund dafür ist, dass bei kleinen Siebintervallen die Wahrscheinlichkeit, zwei Kongruenzen mit gleich großem Primfaktor zu finden, noch gering ist. Daher werden zwar viele partielle Kongruenzen zur „Large Prime Variation“ zwischengespeichert (insgesamt 14479), in den Intervallen der Länge $L = 10000000$ aber nur wenige mehrfache Vorkommen gefunden (insgesamt 263). Bei größeren Intervallen besagt jedoch das sog. „Geburtstagsparadoxon“, dass die Wahrscheinlichkeit, zwei Kongruenzen mit gleichem Primfaktor zu finden, ansteigen wird. In der Tat werden in einem einzigen Intervall der Länge $L = 50000000$ von den 8156 gleichzeitig

ausgewerteten partiellen Kongruenzen 798 kombiniert. Die zwei weiteren Zeilen zeigen, dass bei größerem Annahmeintervall für beide Intervalllängen wie zu erwarten mehr partielle Kongruenzen zur Auswertung zwischengespeichert werden, die Wahrscheinlichkeit jedoch immer weiter sinkt, mehrfache Kongruenzen mit noch größeren Primfaktoren zu finden, sodass nur unwesentlich mehr Kongruenzen kombiniert werden können. Die Dauer von 17.297 Sekunden zur Faktorisierung von $n_{40}$ mit allen zuvor vorgestellten Methoden incl. „Large Prime Variation“ ist gleichzeitig die schnellste Zeit, die mit dem Programm zur Arbeit mit einem Client erreicht wurde.

## 4.5 Ergebnisse der Faktorisierung mit einem Client

Die schnellsten Ergebnisse der Faktorisierungszeiten von Zahlen verschiedener Stellenanzahl mit einem Client und optimalen Einstellungen aller Parameter in Sieb- und Matrixschritt sind in der folgenden Tabelle dargestellt:

| Stellen | Siebschritt | Matrixschritt | Matrixgröße |
|---|---|---|---|
| 20 | 00:00:00.250 | 00:00:00.078 | 76 x 77 ($\approx$ 1KB) |
| 30 | 00:00:01.531 | 00:00:00.125 | 302 x 303 ($\approx$ 11KB) |
| 40 | 00:00:17.297 | 00:00:00.343 | 1388 x 1389 ($\approx$ 235KB) |
| 50 | 00:08:31.187 | 00:00:05.569 | 5220 x 5221 ($\approx$ 3.3MB) |

Besonders der optimierte Matrixschritt mit „xor“-Operationen zeigt, dass selbst Matrizen mit ca. 27 Mio. Einträgen wie der zur 50-stelligen Zahl in Sekunden gelöst werden können. Der Matrixschritt wurde auf einem Intel Core 2 Duo E6300, 1.87 GHz, 2037MB RAM, Betriebssystem Windows Vista, durchgeführt. Als Vergleich wurde die im gesamten Kapitel benutze Testzahl $n_{40}$ zudem auf dem gleichen Client-Computer, auf dem auch die obigen Siebschritte durchgeführt wurden, mit einer Variation verschiedener probabilistischer Faktorisierungsalgorithmen in Derive $^{\text{TM}}$ in 604.2 Sekunden faktorisiert. Bei Zahlen dieser Größenordnung zeigt sich somit bereits die hohe Geschwindigkeit des Quadratischen Siebes gegenüber probabilistischen Algorithmen (für die obige 20-stellige Zahl wurden 0.187 Sekunden, für die 30-stellige Zahl wurden 4.13 Sekunden zur Faktorisierung benötigt).

## 4.6 Parallelisierung

In diesem Abschnitt soll untersucht werden, wie sich im Quadratischen Sieb eine Parallelisierung des Siebschrittes auswirkt. Dazu wurden weitere Zahlen größerer Dezimalstellenzahl mit dem Programm zur Arbeit in einem Netzwerk mit einem Server und maximal 10 Clients, Intel Core 2 Duo E6300, 1.87 GHz, 2037MB RAM, Betriebssystem Windows Vista, unter Verwendung verschiedener Parameter faktorisiert. Der Matrixschritt wurde dabei immer auf einem der Client-Computer durchgeführt:

$$\begin{aligned} n &= 259499077861257819854586300963224352119229541087 73 \\ &= 456874506874568745684508 7 * 567987650780657856507877 9 \end{aligned}$$

| Stellen | 50 | 50 |
|---|---|---|
| Anzahl der Clients | 1 | 10 |
| Smoothness-Bound | 109601 | 109601 |
| Siebschritt | 00:12:15.766 | 00:01:25.547 |
| Matrixschritt | 00:00:05.569 | 00:00:05.569 |
| Matrixgröße | 5220 x 5221 ($\approx$ 3.3MB) | 5220 x 5221 ($\approx$ 3.3MB) |

Die angegebene 50-stellige Zahl wurde mit heuristischen Parametern auf einem und auf zehn Clients faktorisiert. Dabei zeigt sich, dass der Siebschritt auf zehn Computern verteilt auch ca. 10 Mal schneller durchgeführt wird. Dividiert man die 12 Minuten 15 Sekunden für einen Client durch 10, so hätten 10 Clients 1 Minute 14 Sekunden für die Zahl benötigen sollen. Die wahre Laufzeit ist etwas länger, da bei der benutzen Intervalllänge von $L = 20000000$ nicht alle Clients sofort anhielten, sobald beim Server ausreichend Kongruenzen eingegangen waren, sondern sich erst nach Abschließen ihres letzten Intervalls beim Server meldeten.

$$\begin{aligned} n &= 157960946069428945351698163127485492713793693851463360012417 \\ &= 345763495763457364597635763619 * 456846798476067846787273487243 \end{aligned}$$

| Stellen | 60 | 60 |
|---|---|---|
| Anzahl der Clients | 10 | 10 |
| Smoothness-Bound | 200000 | 417367 |
| Siebschritt | 00:30:42.828 | 00:13:08.594 |
| Matrixschritt | 00:00:42.354 | 00:01:46.222 |
| Matrixgröße | 8968 x 8870 ($\approx$ 9.5MB) | 17609 x 17610 ($\approx$ 37.0MB) |

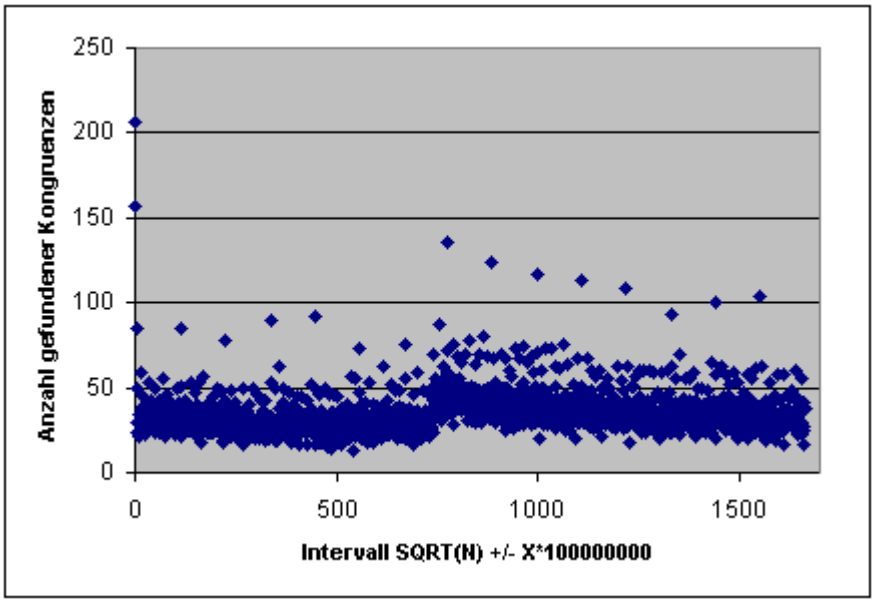
Die 60-stellige Zahl wurde auf 10 Clients mit zwei verschiedenen Parametern für die Smoothness-Bound faktorisiert. Im ersten Fall wurde die Smoothness-Bound manuell auf $B = 200000$ korrigiert, um eine Matrix der Dimension von ca. 9000x9000 zu erhalten. In diesem Fall enthielt die Faktorbasis nur 8968 Primzahlen, sodass nur wenige glatte Kongruenzen gefunden wurden und der Siebschritt entsprechend lange dauerte. Dafür konnte die kleinere Matrix schnell gelöst werden. Im zweiten Fall wurden der heuristische Parameter für die Smoothness-Bound benutzt. Dadurch wurden mit 17609 Primzahlen in der Faktorbasis schneller alle Kongruenzen gefunden, die Matrix enthielt dafür jedoch auch ca. vier Mal so viele Einträge, wodurch der Matrixschritt länger dauerte. Insgesamt zeigt sich aber, dass die Faktorisierung mit heuristischer Smoothness-Bound und einer Gesamtzeit von 14 Minuten 54 Sekunden ca. doppelt so schnell wie die mit kleiner Matrix und Gesamtzeit 31 Minuten 25 Sekunden durchgeführt wird.

$$\begin{aligned} n &= 259416389801187537794190049105739376862110587863755415601322078279352 1 \\ &= 456874568749567456794567459674569 * 567806587508670865750867560875687 59 \end{aligned}$$

| Stellen | 70 |
|---|---|
| Anzahl der Clients | 10 |
| Smoothness-Bound | 1527397 |
| Siebschritt | 03:57:02.672 |
| Matrixschritt | 00:37:58.691 |
| Matrixgröße | 58355 x 58375 ($\approx$ 406MB) |

Die größte im Rahmen der Arbeit faktorisierte Zahl hatte 70 Stellen. Als Smoothness-Bound wurde der heuristische Wert benutzt, für den der Siebschritt in ca. 4 Stunden auf 10 Clients abgeschlossen wurde. Die Berechnung der Faktoren im Matrixschritt per Gauß-Jordan-Algorithmus auf den ca. 400MB Daten verlief dabei in 38 Minuten relativ schnell.

Für die obige 70-stellige Zahl zeigt das folgende Diagramm die Verteilung der insgesamt gefundenen Kongruenzen über und unter der Wurzel. Bis zum Ende des Siebschrittes wurden 1663 Intervalle der Länge $L = 100000000$ bearbeitet.




Wie im vorherigen Diagramm ist zu sehen, dass die Anzahl ab den 206 Kongruenzen im ersten Intervall sofort abfällt, sich aber dann in den über 1500 folgenden Intervallen beim Mittelwert von 35 Kongruenzen pro Intervall stabilisiert. Bis zum Abschluss der Siebschrittes werden jedoch regelmäßig auch Intervalle mit ca. 100 Kongruenzen gefunden.

## 4.7 Theoretische Laufzeit

Die theoretische Laufzeit des Quadratischen Siebes beträgt nach [1]

$$O\left(\exp\left(\sqrt{\ln n \ln\ln n}\right)\right).$$

Eine Herleitung dieser Abschätzung ist ebenfalls in [1] zu finden. Exemplarisch kann die Formel für den Siebschritt der 50- und 70-stelligen Zahl mit 10 Clients überprüft werden. Nach der Abschätzung benötigt das Quadratische Sieb für eine 70- statt einer 50-stelligen Zahl die

$$\exp\left(\sqrt{\ln 10^{70} \ln\ln 10^{70}}\right) / \exp\left(\sqrt{\ln 10^{50} \ln\ln 10^{50}}\right) \approx 190$$

fache Zeit. Rechnet man mit der zuvor diskutierten Zeit von 1 Minute 14 Sekunden für 50 Stellen so ergibt sich eine Laufzeit von 3.91 Stunden für 70 Stellen. Dies stimmt mit der gemessenen Zeit von 3.95 Stunden (dies entspricht 3 Stunden 57 Minuten) für die faktorisierte Zahl mit 70 Stellen gut überein.

# 5 Diskussion der Ergebnisse

Ziel der Arbeit war die Programmierung eines leistungsfähigen Netzwerks zur parallelisierten Faktorisierung von ganzen Zahlen mit dem Quadratischen Sieb. Es stellte sich heraus, dass mit dem Programm eine Faktorisierung von Zahlen bis 50 Stellen problemlos auf einem Client-Computer in Sekunden oder Minuten durchführbar ist, parallelisiert im Zeitraum von Minuten oder Stunden auch 60- und 70-stellige Zahlen zerlegt werden können. Dabei wurde deutlich, dass die Verwendung verschiedener Parameter starke Auswirkungen auf die Laufzeit der Faktorisierung hat und die heuristischen Werte meist die besten Resultate lieferten. Zudem wird eine Faktorisierung großer Zahlen mit der Standardvariante des Quadratischen Siebes nur mit Hilfe aller in der Arbeit vorgestellten Optimierungsmethoden wie der „Large Prime Variation“ möglich. Durch Parallelisierung mit $m$ Computern statt einem kann die Laufzeit um ca. einen Faktor $m$ verkürzt werden.

Eine Kapazitätsbeschränkung des Quadratischen Siebes stellt der Matrixschritt dar, der auf einem Computer mit großen Matrizen durchgeführt werden muss. Obwohl der Matrixschritt im Programm zur Arbeit sehr schnell läuft, kann das Programm um schnellere Matrixalgorithmen erweitert werden um Speicherplatz und Rechenaufwand zu minimieren und so Faktorisierungen von Zahlen mit mehr als 70 Stellen zu ermöglichen. Eine Erweiterung auf das sog. „Multiple Polynomial Quadratic Sieve“ (MPQS) (siehe Ausblick im 6. Kapitel) wäre zudem sinnvoll.

# 6 Ausblick auf Erweiterungen des Basisalgorithmus des Quadratischen Siebes

Der Basisalgorithmus des Quadratischen Siebes, wie er in dieser Arbeit beschrieben wurde, ist seit der Entwicklung im Jahr 1981 von Carl Pomerance weiter verbessert worden. Wichtigste Erweiterungen sind das „Multiple Polynomial Quadratic Sieve“ (MPQS), unabhängig vorgestellt von Davis, Holdridge und Montgomery und das „Number Field Sieve“ (NFS). In der MPQS-Variante wird das Polynom $f(x) = x^2 - n$, für das $B$-glatte Werte gesiebt werden, durch ein Polynom $g(x) = f(ax + b)$ ersetzt. Die Grundidee ist dabei, dass in der Basisversion des Quadratischen Siebes die Anzahl der Kongruenzen, die über der Faktorbasis zerfallen, ab der Wurzel stark abnimmt. Aus diesem Grund wird statt $x$ ein lineares Polynom $ax + b$ verwendet, dessen Koeffizienten $a$ und $b$ zusätzliche Bedingungen erfüllen müssen. Vorteil der Methode ist, dass im Siebschritt ein festes Siebintervall gewählt werden kann. Nach Abschluss des Intervalls wird der Siebbereich nicht vergrößert, sondern ein neues Polynom gewählt und das gleiche Intervall mit dem nächsten Polynom erneut gesiebt. Auf diese Weise werden mehr $B$-glatte Kongruenzen gefunden. Eine heuristische Abschätzung nach [1] zeigt, dass die Basisversion des Quadratischen Siebes so um einen Faktor $\frac{1}{2}\sqrt{\ln n \ln\ln n}$ beschleunigt wird. Für eine Zahl mit 100 Stellen bedeutet dies eine ca. 17-fache Beschleunigung des Basisalgorithmus.

Wichtigste Erweiterung des Quadratischen Siebes ist das Zahlkörpersieb („Number Field Sieve“, NFS). Dieses Verfahren existiert in einer „speziellen“ und „allgemeinen“ Variante. Das spezielle Zahlkörpersieb wurde 1988 von J. M. Pollard erstmals beschrieben und von A. K. Lenstra, H. W. Lenstra, M. S. Manasse und J. M. Pollard in den Folgejahren verbessert. Das Verfahren ist nur auf Zahlen der Form $b^m - r$ mit $b, r, m \in \mathbb{N}$ und $b, r << m$ anwendbar. Das allgemeine Zahlkörpersieb wurde von J. P. Buhler, H. W. Lenstra und C. Pomerance entwickelt und eignet sich auch zur Faktorisierung von Zahlen ohne spezielle Struktur, ist dafür allerdings langsamer. Das Zahlkörpersieb gilt als das schnellste bekannte Faktorisierungsverfahren für große Zahlen ab ca. 110 Stellen. Wichtigster Erfolg ist die Faktorisierung von RSA-200 (200 Dezimalstellen) in den Jahren 2003 bis 2005, der größten bisher faktorisierten Zahl aus zwei großen Primzahlen ohne spezielle Struktur.

# 7 Eigenständigkeitserklärung

Ich erkläre, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Datum Unterschrift

# Literatur


[1] R. Crandall, C. Pomerance, *Prime Numbers – A Computational Perspective*, Springer-Verlag, 2001

[2] H. Riesel, *Prime Numbers and Computer Methods for Factorization*, Second Edition, Birkhäuser, 1994

[3] B. Hoffmann, *Faktorisierung großer Zahlen mit dem Quadratischen Sieb*, Studienarbeit Nr. 2007, Universiät Stuttgart, Institut für Formale Methoden der Informatik

[4] I. Storey, *Quadratic Sieve Factorisation on a Network*, RMIT University in the School of Business Information Technology (VET), Melbourne, Australia, 2003

[5] E. Landquist, *The Quadratic Sieve Factoring Algorithm*, MATH 488: Cryptographic Algorithms, 2001

[6] M. Martin, *NX - Numerics Library*, www.ellipsa.net, 2007

[7] Wikipedia, Artikel *Quadratisches Sieb*, *Zahlkörpersieb*, de.wikipedia.org

[8] C. Lanczos, *Solution of systems of linear equations by minmalized iterations*, J. Res. Nat. Bur. Standards, 1952

[9] D. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Trans. Inform. Theory, 1986